LA-UR-21-29468

Title:	LUNA Condition Based Monitoring Update: Random Forest and Mahalanobis Ensemble Accuracy Crossover Point
Author(s):	Green, Andre Walter
Intended for:	Progress report to sponsor.
Issued:	2021-09-27

LUNA Condition-Based Monitoring Update: Random Forest and Mahalanobis Ensemble Accuracy Crossover Point

Presented 9/8/2021

Andre Green

Feature Performance for each Dataset

[Mahalanobis Ensemble v. Random Forest]

An isolation forest was used to remove potentially anomalous/outlier points (points possibly between changing between damage type/severities); approximately 10% for Board 401 and Multi-Actuator, 20% for Philadelphia dataset.

Uniformly Selected* Features

[Variable number of features]

1 out of 100 samples was used. This is the same 'feature vector' which was used in the previous report for the random forest.

Random Forest

~92% for Board 401 Dataset

~90% for Multi-Actuator Dataset

~100% for Philadelphia Dataset

Mahalanobis Ensemble

The covariance matrix was always singular, so the Mahalanobis ensemble doesn't work for this set of features (on any of the datasets).

Ali's Features [13 features]

['Var_of_Accel_1', 'Var_of_Accel_2', 'Var_of_Accel_3', 'Mean_of_PG_1', 'Mean_of_PG_2', 'Mean_of_PG_3', 'Var_of_PG_1', 'Var_of_PG_2', 'Var_of_PG_3', 'Slope_of_Angle', 'Pressure_Diff_Sum', 'Diff_Temp_Var', 'Pressure_Max']

Random Forest

~86% for Board 401 Dataset

~75% for Multi-Actuator Dataset

~98% for Philadelphia Dataset

Mahalanobis Ensemble

~87% for Board 401 Dataset

~83% for Multi-Actuator Dataset

~99% for Philadelphia Dataset

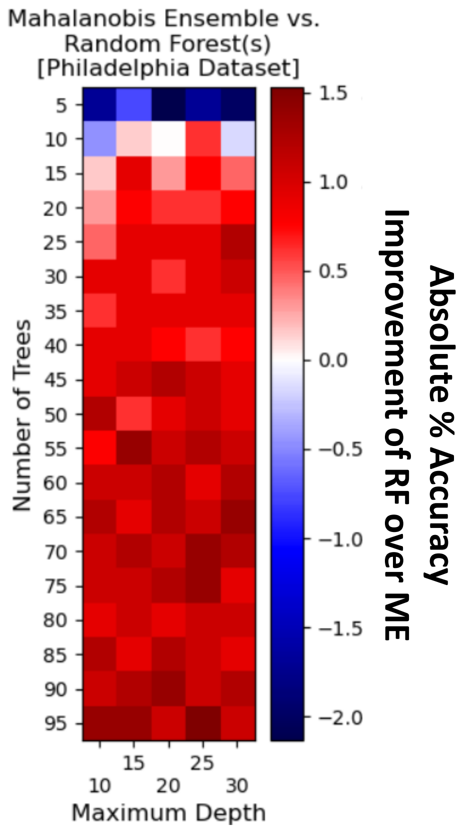
The random forest of size 16 with maximum depth 10 does comparably well (the means, mins, maxes, and medians of accuracy across the 9 folds are similar) to the Mahalanobis Ensemble when used with Ali's features.

The random forest gets better performance using the uniformly-down-sampled 'features' (1 out of every 100 samples) regardless of the dataset: the only downside is the results may not be as easily interpretable.

Performance Maps for each Dataset [Using Ali's Features]

[Mahalanobis Ensemble v. Random Forest] **% Improvement of RF over ME shown by color.**

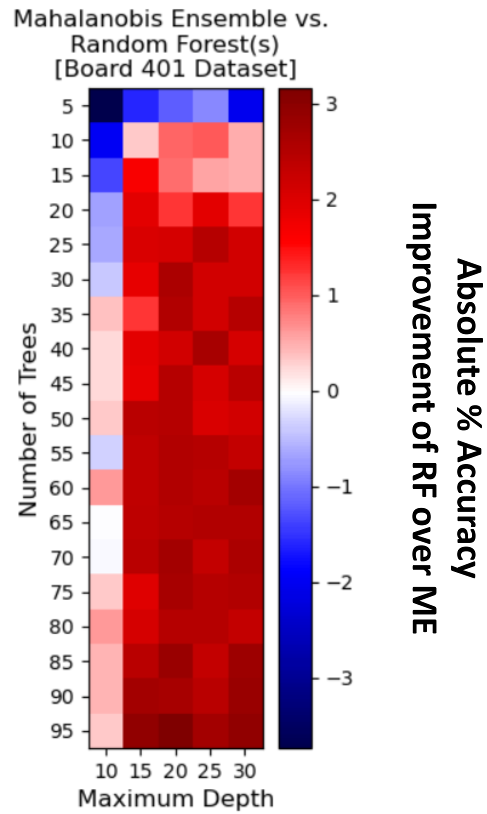
Philadelphia Dataset(s)



Dataset(s):

philadelphia_9_10_19
philadelphia_9_11_19_Act_1
philadelphia_9_11_19_Act_2
philadelphia_9_11_19_Act_5
philadelphia_9_11_19_Act_6

Board_401 Dataset

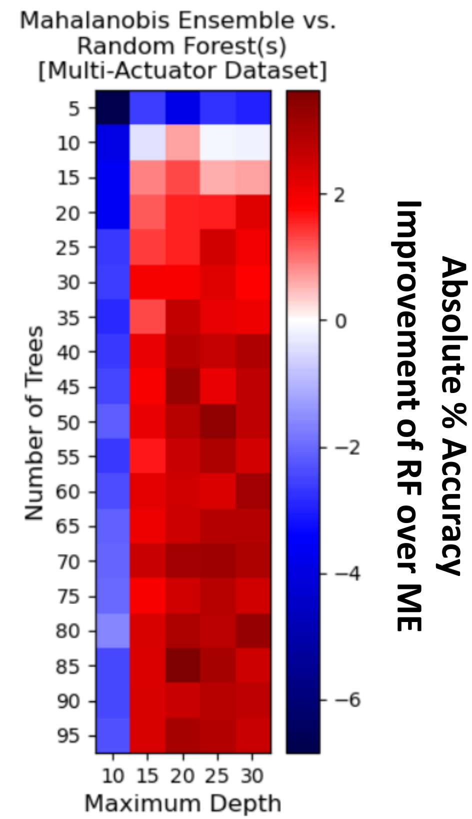


Dataset(s):

ali (Board_401)

RF **worse** than ME
RF **better** than ME

Multi-Actuator Dataset(s)



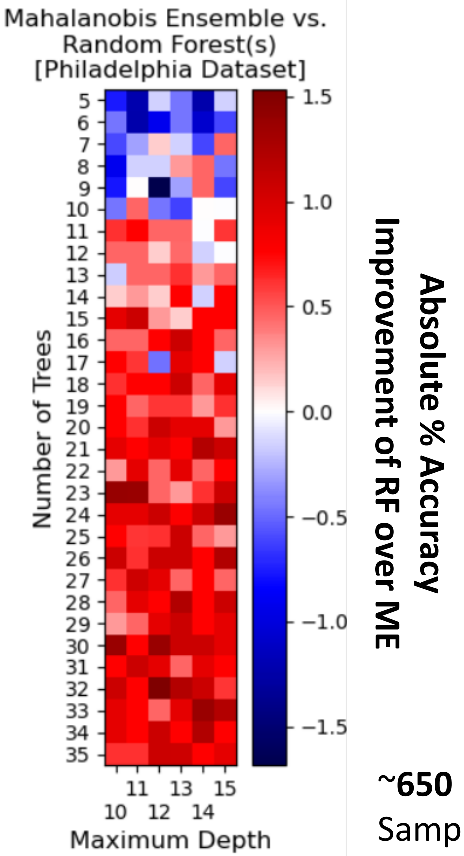
Dataset(s):

25K_Cycles
51.4K_Cycles
101K_Cycles

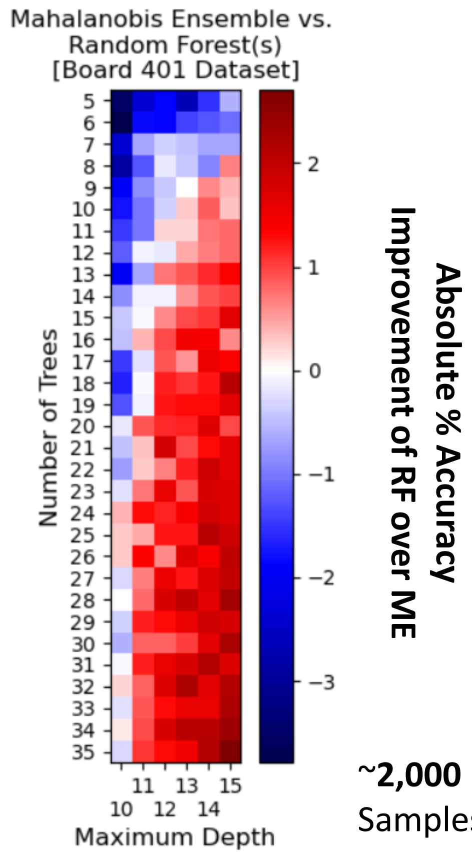
Performance Maps for each Dataset [Using Ali's Features] [Detail]

[Mahalanobis Ensemble v. Random Forest] **% Improvement of RF over ME shown by color.**

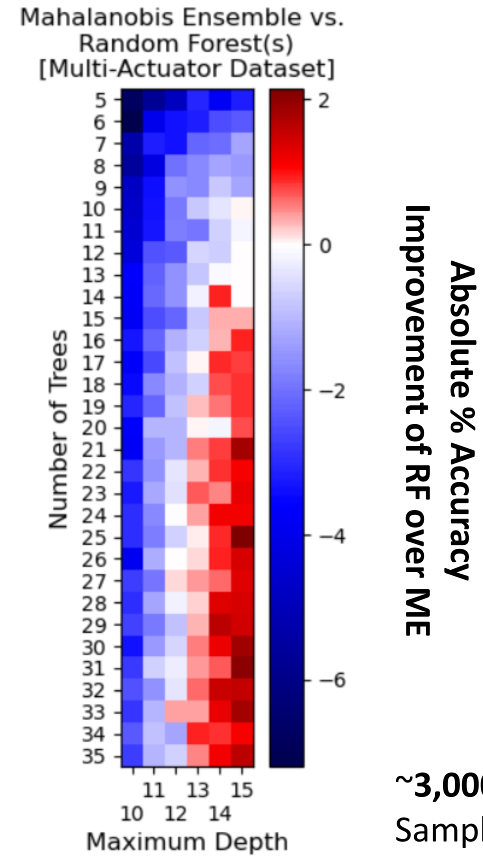
Philadelphia Dataset(s)	**Board_401 Dataset**	**Multi-Cycle Dataset(s)**



~650
Samples



~2,000
Samples



~3,000
Samples

The accuracy crossing point seems to be around **10 trees with a depth of 12**. Having either fewer trees or less deep trees for the same amount of depth or number of trees results in worse performance than the Mahalanobis Ensemble. However, for the multi-cycle dataset, it seems slightly more trees/greater depth are required for the RF to perform as well as the ME (14+ trees, depth of 13-14+, [>106,496 numbers]).

(10 * 2^12) = 40,960 numbers, if the trees are all densely populated (using 10 trees with a max depth of 12)

(21 * 13^13) = 46,137 numbers, when using 21 classes and 13 features.

Time & Space Complexity

Mahalanobis Ensemble

For C classes and F-dimensional feature vectors:

Mahalanobis Ensemble scales with the **number of features**.

Time complexity: O(C * (F^3)) C [FxF] matrix multiplications.
Space complexity: O(C * (F^2)) C [FxF] matrices.

Random Forest

For T trees with maximum depth D:

Random Forest scales **with number of trees** and **max depth of trees**.

Time complexity: O(T * D) T traversals of D-deep trees.
Space complexity: O(T * 2^D) T D-deep trees.

If the random forest is checking multiple variables (say m) at each node of its trees, then the time & space complexities just change linearly: O(m * T * D) for time, O(m * T * 2^D) for space.